Programming Engineering

Course 4 – 15 March 2017

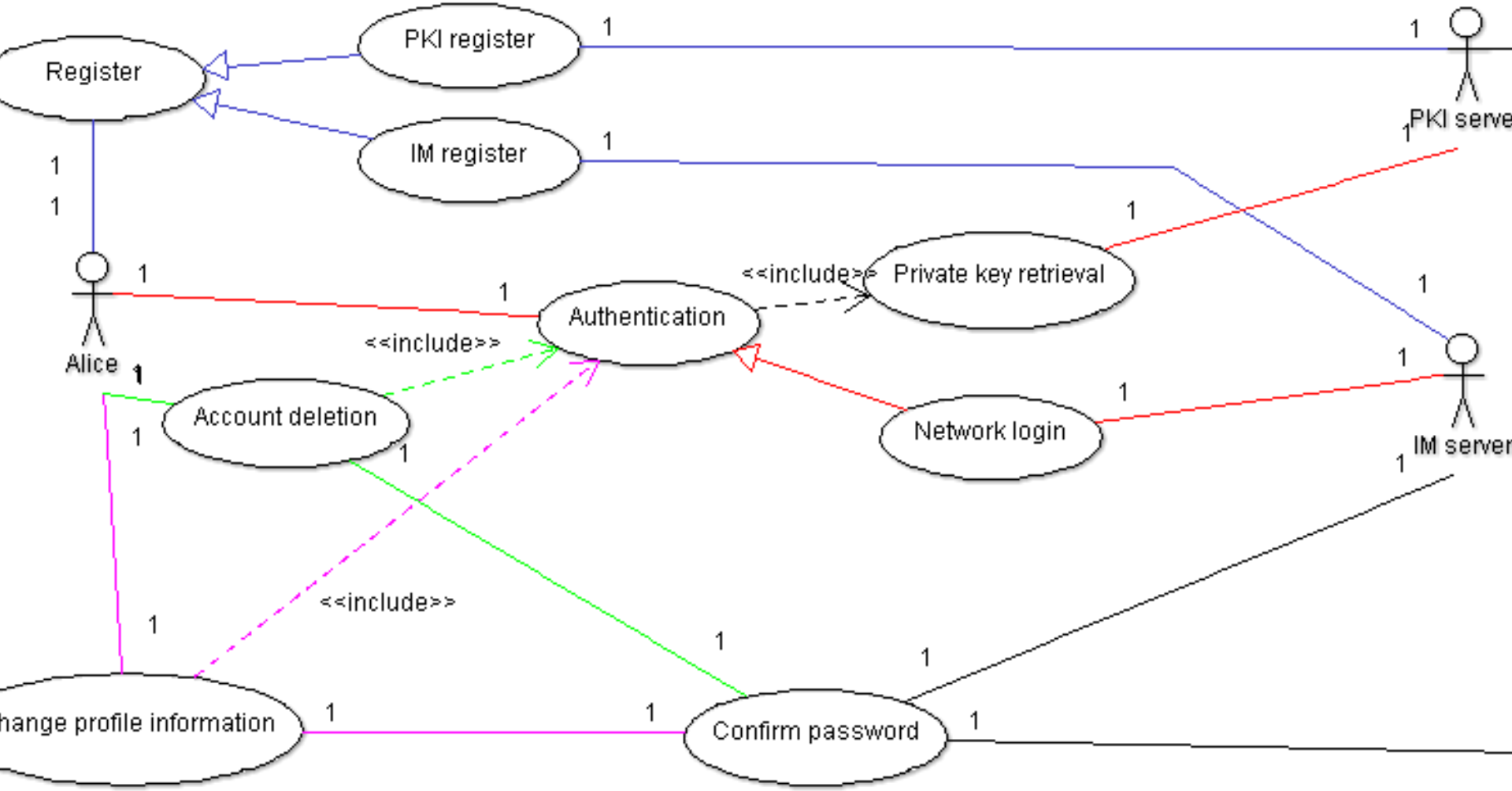
adiftene@info.uaic.ro

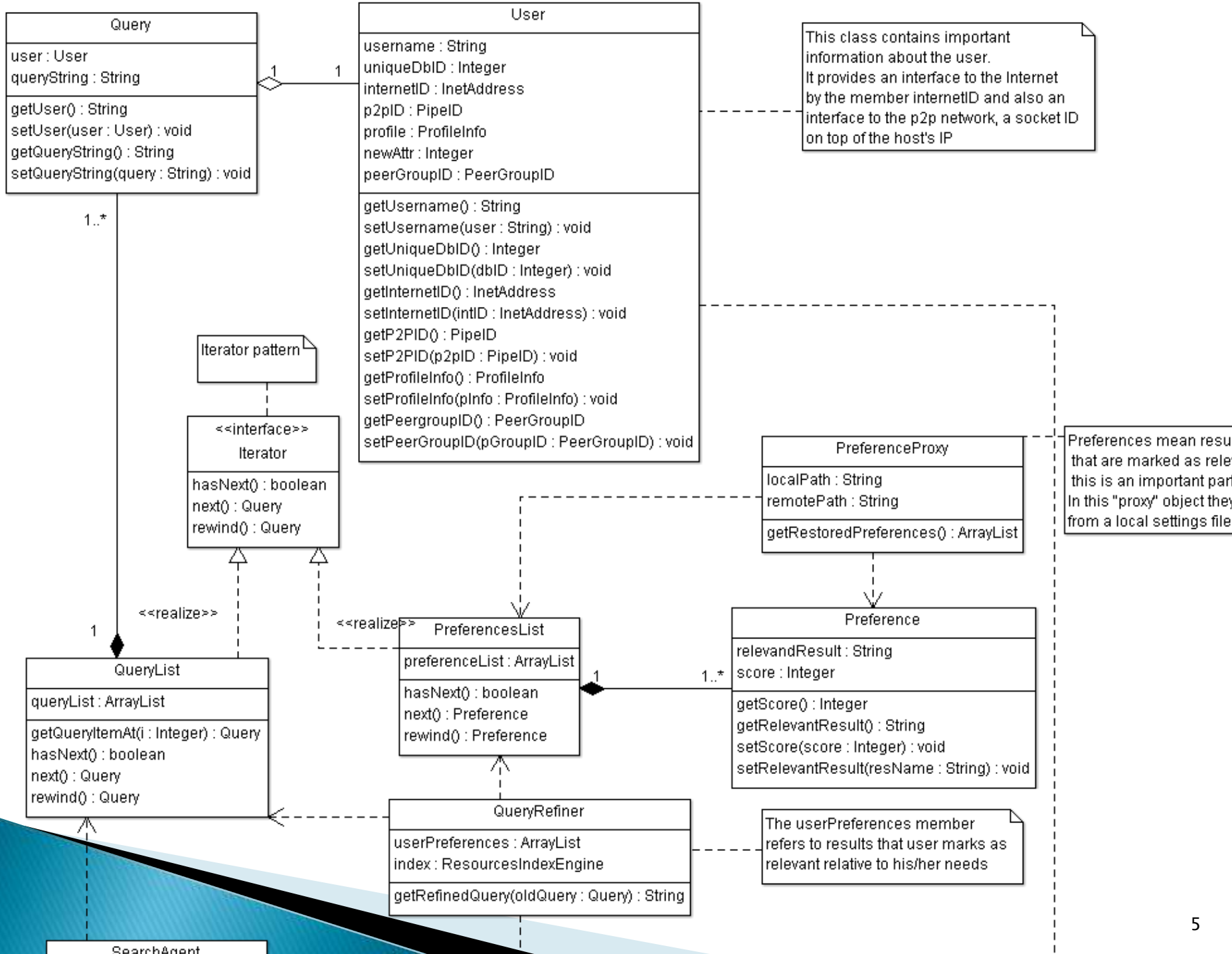
Content

- From previous courses…
- UML Diagrams:
  - Interactions (Sequence, Collaboration)
  - Behaviour (States, Activities)
  - Structure (Deployment)

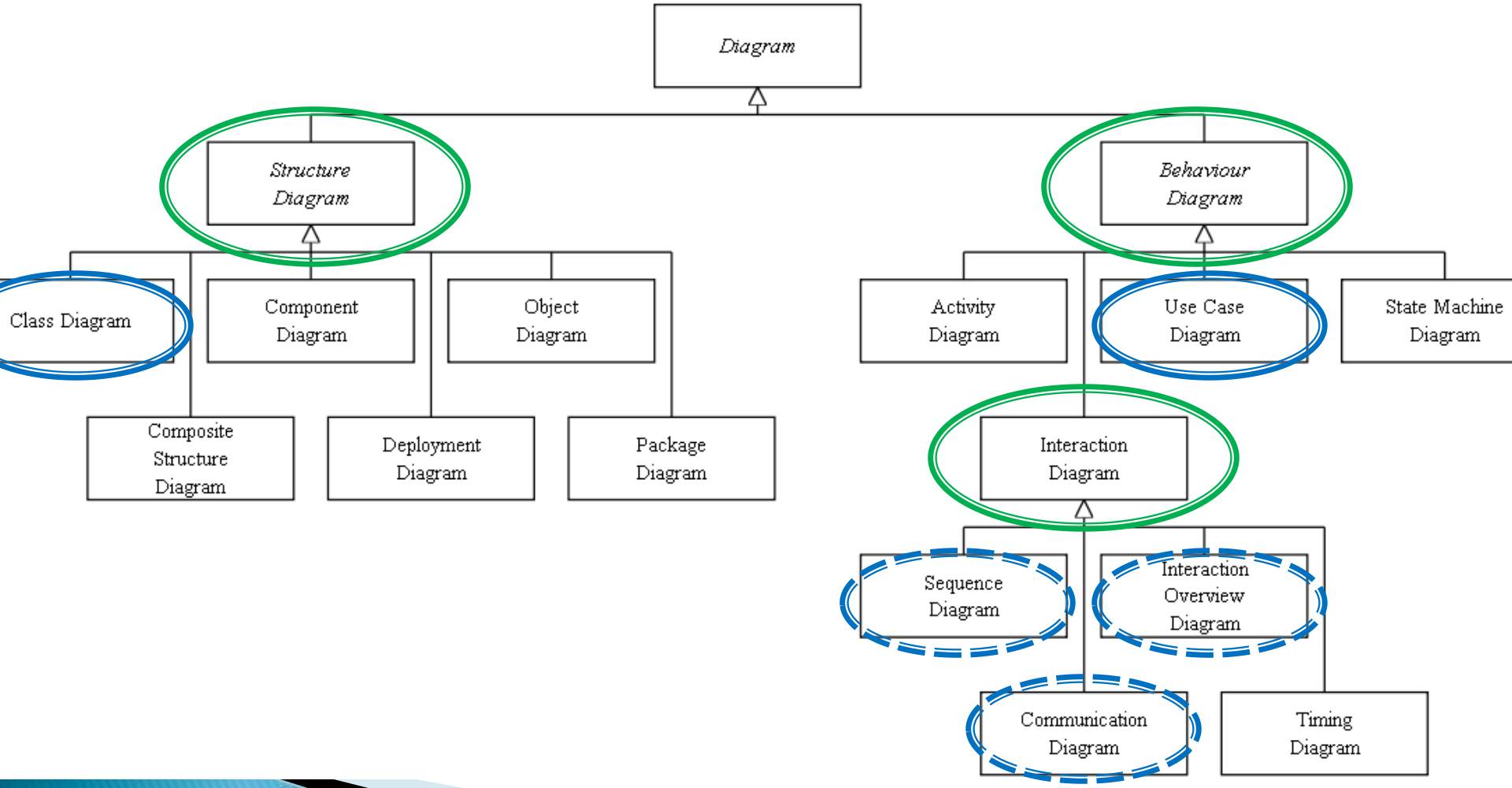
From previous courses…

- Diagrams
- UML Diagrams
- Use Case Diagrams
- Class Diagrams


Register
PKI register
IM register
PKI serve
Alice
Authentication
<<include>>
Private key retrieval
Account deletion
Network login
IM server
<<include>>
hange profile information
Confirm password
1

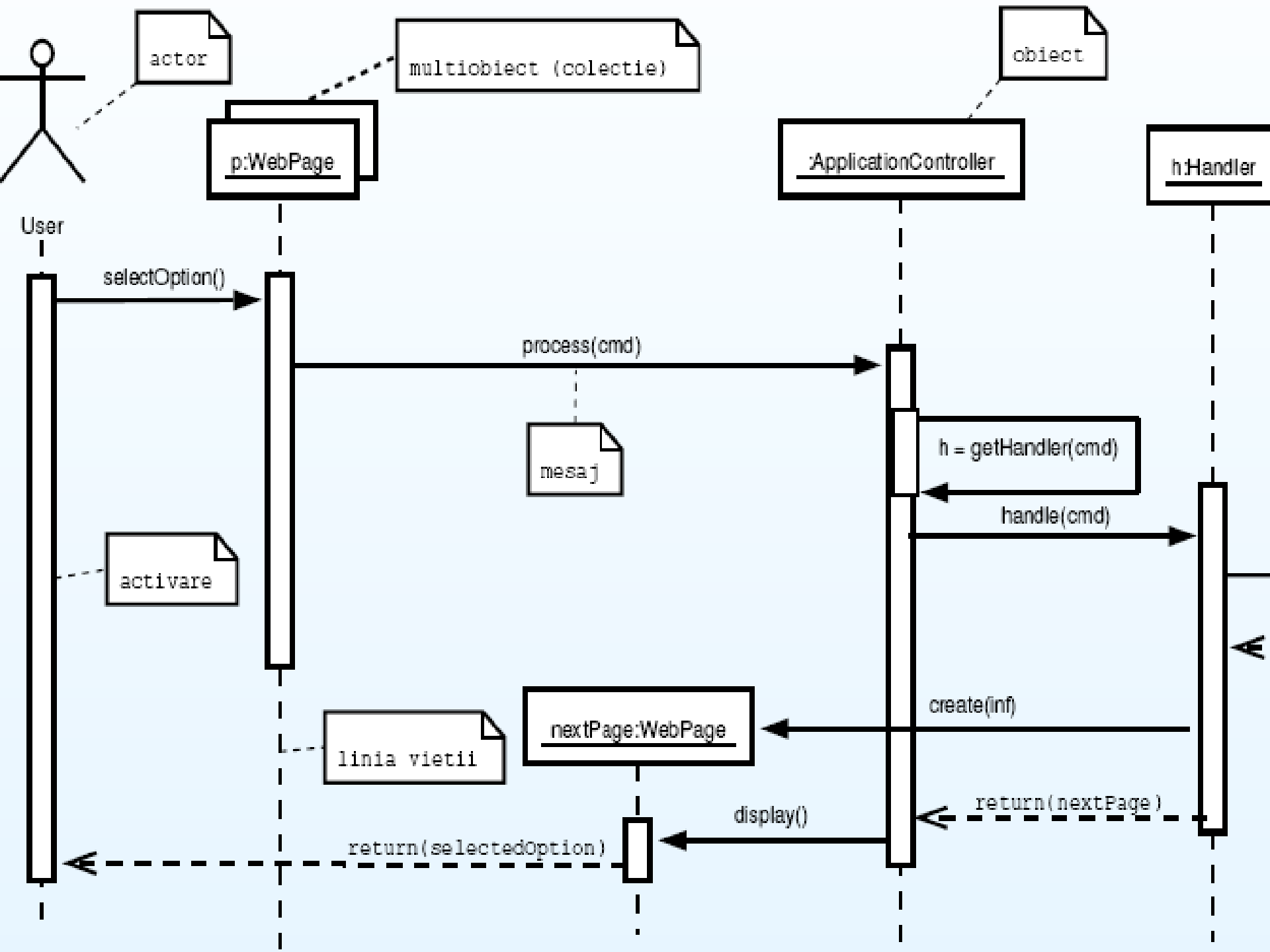

Query
user : User
queryString : String
getUser() : String
setUser(user : User) : void
getQueryString() : String
setQueryString(query : String) : void
1
1
User
username : String
uniqueDbID : Integer
internetID : InetAddress
p2pID : PipeID
profile : ProfileInfo
newAttr : Integer
peerGroupID : PeerGroupID
getUsername() : String
setUsername(user : String) : void
getUniqueDbID() : Integer
setUniqueDbID(dbID : Integer) : void
getInternetID() : InetAddress
setInternetID(intID : InetAddress) : void
getP2PID() : PipeID
setP2PID(p2pID : PipeID) : void
getProfileInfo() : ProfileInfo
setProfileInfo(pInfo : ProfileInfo) : void
getPeergroupID() : PeerGroupID
setPeerGroupID(pGroupID : PeerGroupID) : void
This class contains important information about the user.
It provides an interface to the Internet by the member internetID and also an interface to the p2p network, a socket ID on top of the host's IP
1..*
Iterator pattern
<<interface>>
Iterator
hasNext() : boolean
next() : Query
rewind() : Query
PreferenceProxy
localPath : String
remotePath : String
getRestoredPreferences() : ArrayList
Preferences mean resu
that are marked as rele
this is an important part
In this "proxy" object they
from a local settings file
<<realize>>
<<realize>>
1
PreferencesList
preferenceList : ArrayList
hasNext() : boolean
next() : Preference
rewind() : Preference
1
1..*
Preference
relevandResult : String
score : Integer
getScore() : Integer
getRelevantResult() : String
setScore(score : Integer) : void
setRelevantResult(resName : String) : void
QueryList
queryList : ArrayList
getQueryItemAt(i : Integer) : Query
hasNext() : boolean
next() : Query
rewind() : Query
QueryRefiner
userPreferences : ArrayList
index : ResourcesIndexEngine
getRefinedQuery(oldQuery : Query) : String
The userPreferences member refers to results that user marks as relevant relative to his/her needs
SearchAgent

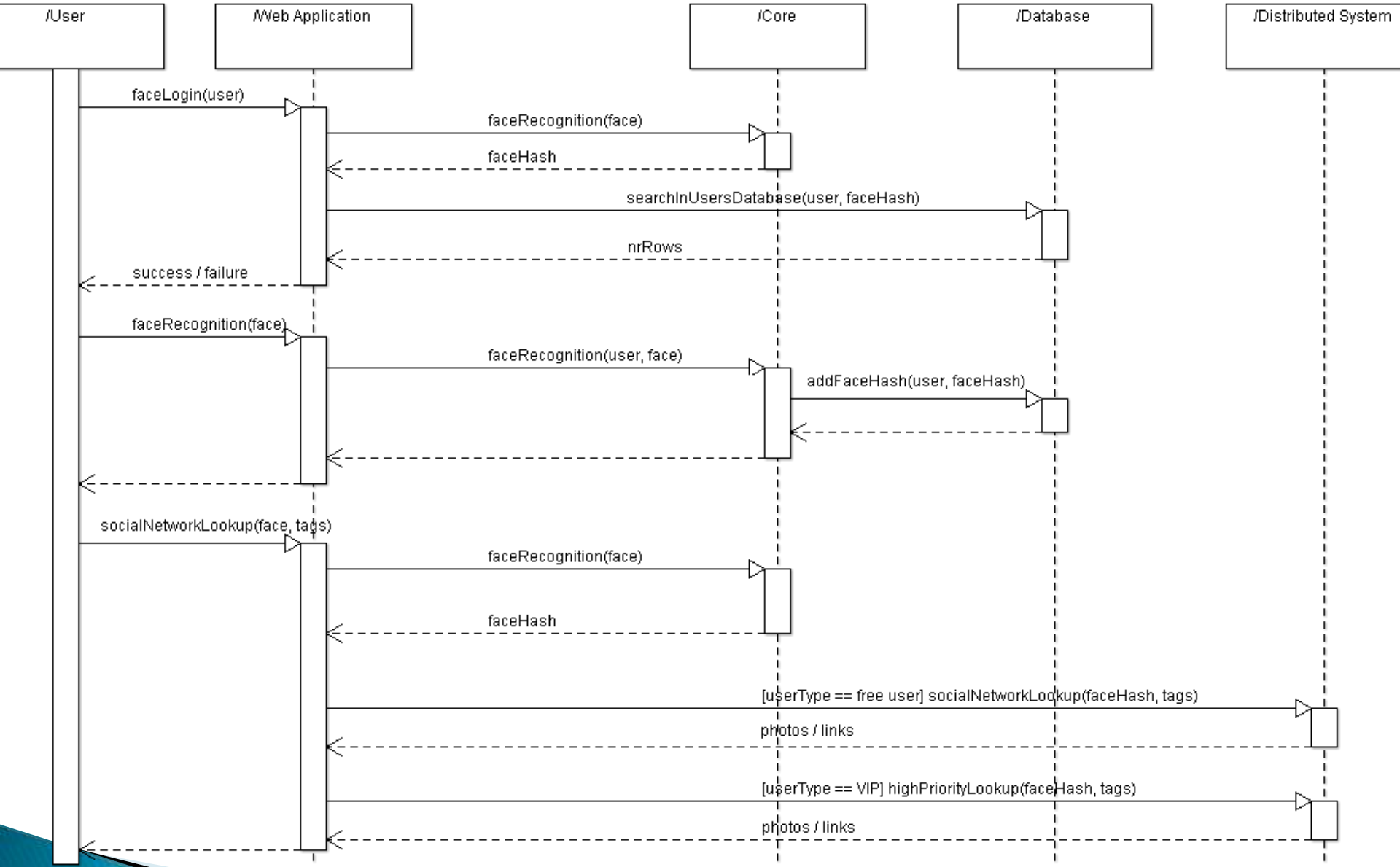
UML2.0 – 13 Types of Diagrams



Sequence Diagram

- **Sequence Diagram** contents the sequence of actions which occur in the system, each object invocation methods, as well as the order in time in which these invocations take place
- A sequence diagram is bi-dimensional
  - On the vertical axis it shows the life of the object
    - The lifeline of the objects (graphic: dotted line)
    - The activation period in which an object takes control of execution (graphic rectangle on the lifeline)
  - The horizontal axis shows the sequence of creation or the invocations
    - messages ordered in time (graphic: darts)

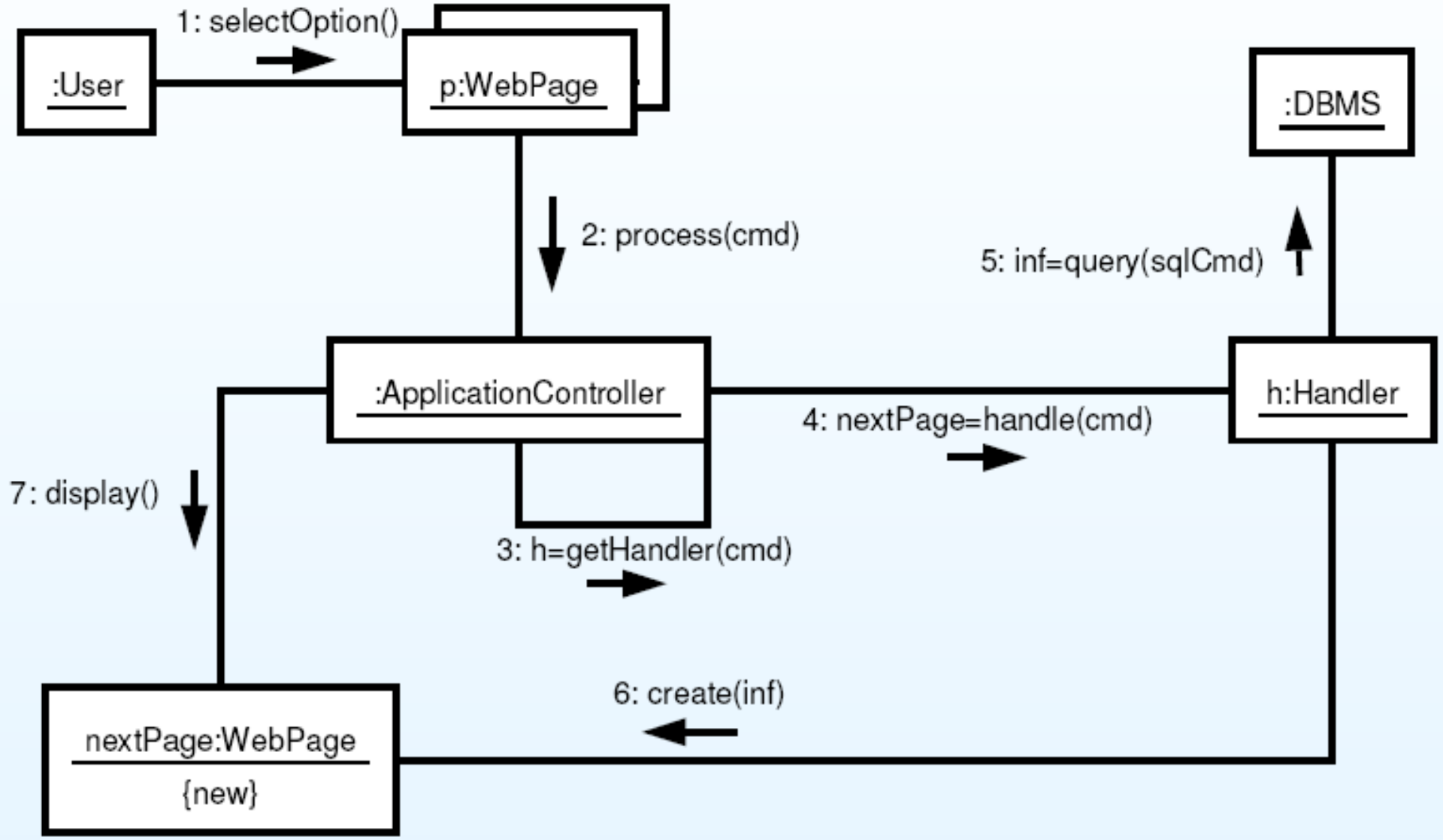

actor
multiobiect (colectie)
obiect
p:WebPage
:ApplicationController
h:Handler
User
selectOption()
process(cmd)
mesaj
h = getHandler(cmd)
handle(cmd)
activare
create(inf)
nextPage:WebPage
linia vietii
return(nextPage)
display()
return(selectedOption)


/User
/Web Application
/Core
/Database
/Distributed System
faceLogin(user)
faceRecognition(face)
faceHash
searchInUsersDatabase(user, faceHash)
nrRows
success / failure
faceRecognition(face)
faceRecognition(user, face)
addFaceHash(user, faceHash)
socialNetworkLookup(face, tags)
faceRecognition(face)
faceHash
[userType == free user] socialNetworkLookup(faceHash, tags)
photos / links
[userType == VIP] highPriorityLookup(faceHash, tags)
photos / links

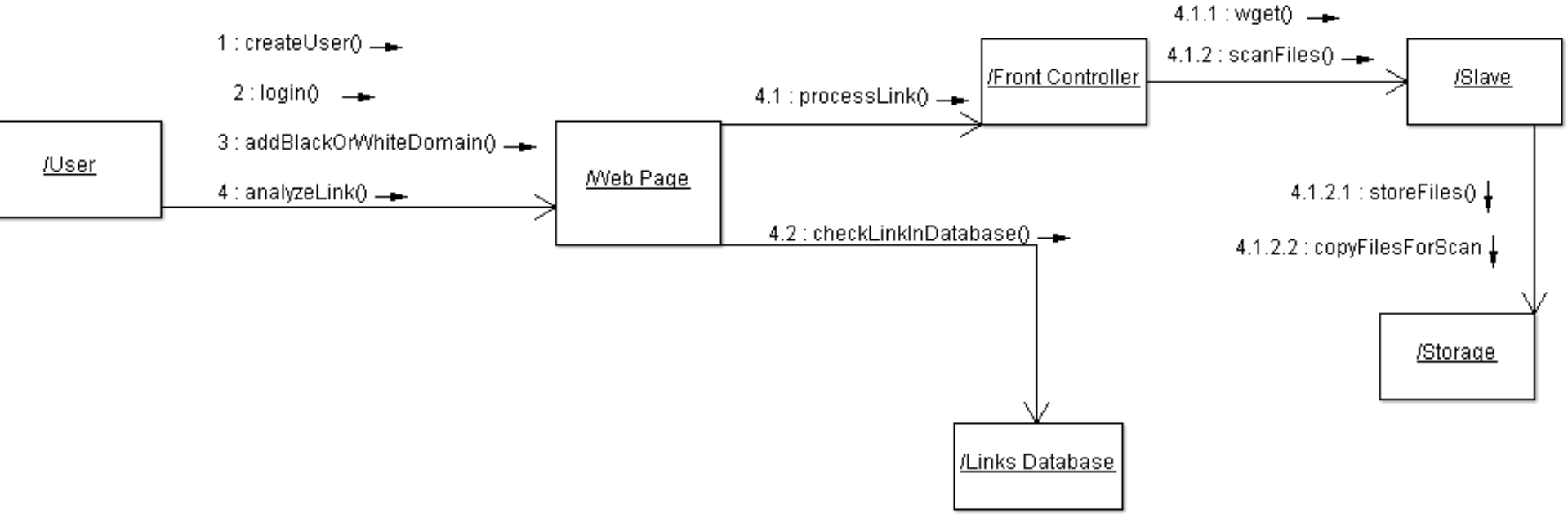
Collaboration Diagram

- Emphasizes the structural organization of the objects participating in the interaction
- It illustrates better complex ramifications, iterations and concurrent behavior
- May contain:
  - Objects, classes, actors
  - Links between them
  - Messaging

Example 1


1: selectOption()
:User
p:WebPage
:DBMS
2: process(cmd)
5: inf=query(sqlCmd)
:ApplicationController
h:Handler
4: nextPage=handle(cmd)
7: display()
3: h=getHandler(cmd)
6: create(inf)
nextPage:WebPage
{new}

Example 2

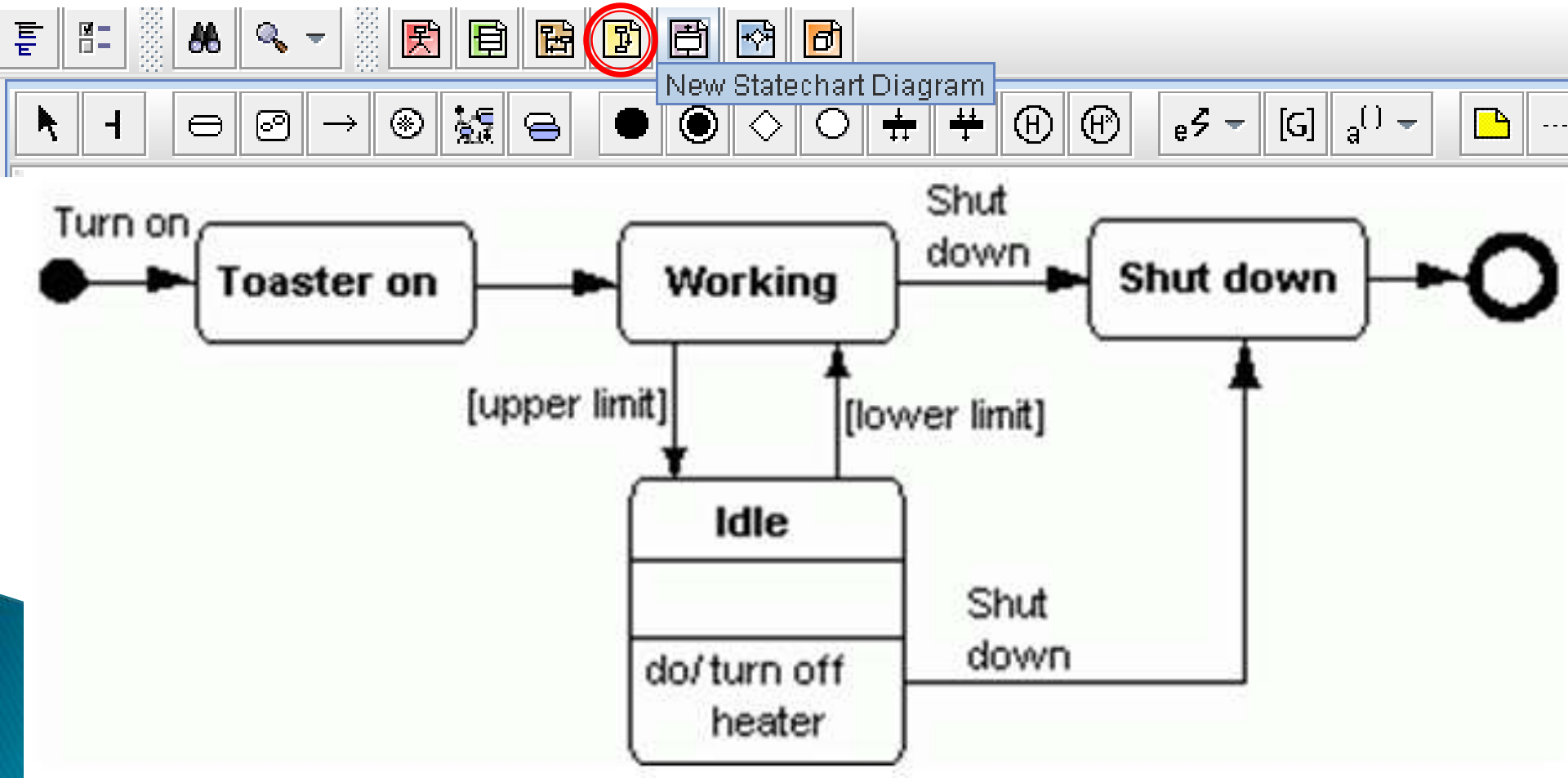

1 : createUser()
2 : login()
3 : addBlackOrWhiteDomain()
4 : analyzeLink()
/User
/Web Page
4.1 : processLink()
/Front Controller
4.1.1 : wget()
4.1.2 : scanFiles()
/Slave
4.1.2.1 : storeFiles()
4.1.2.2 : copyFilesForScan
/Storage
4.2 : checkLinkInDatabase()
/Links Database

Behavioral Diagrams

- State Diagrams, Activity Diagrams
- Basic elements
  - Event
  - Action
  - Activity

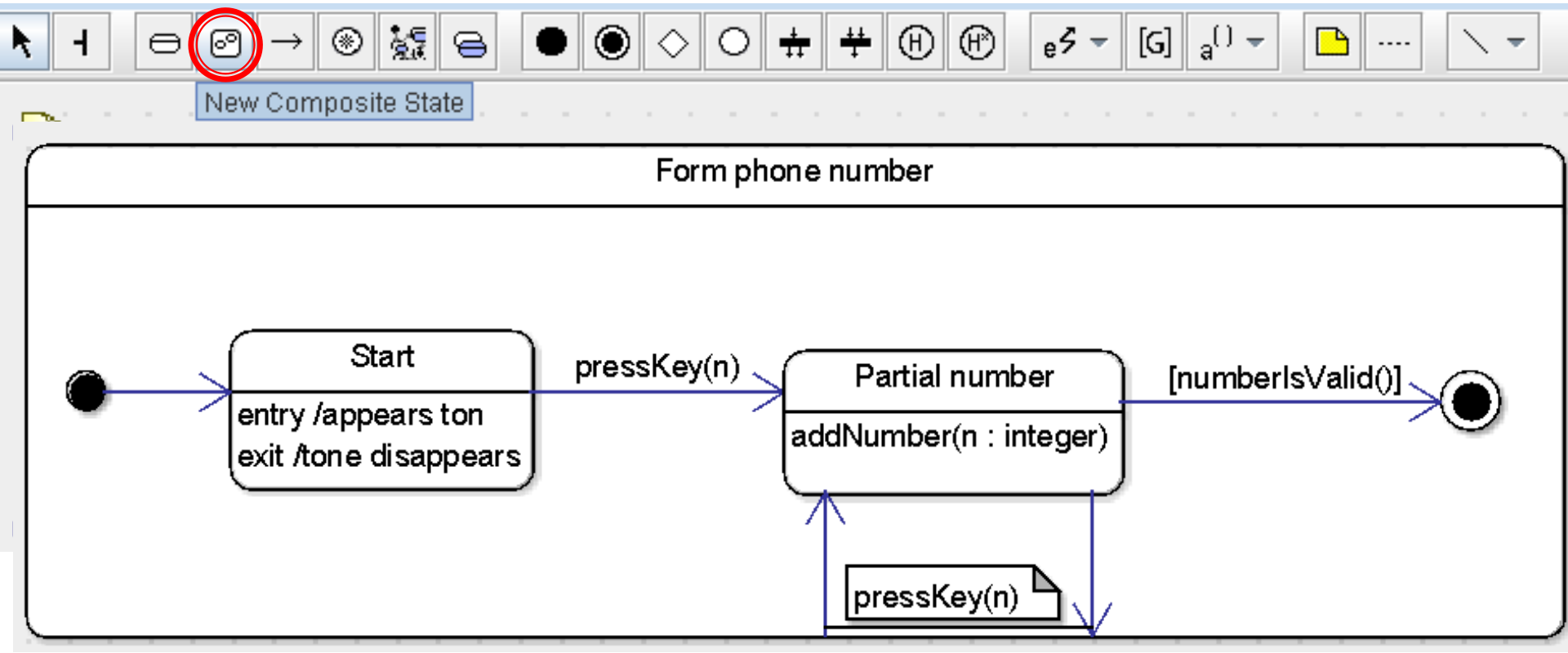
Statechart Diagrams

- Contain:
  - States
  - Transitions



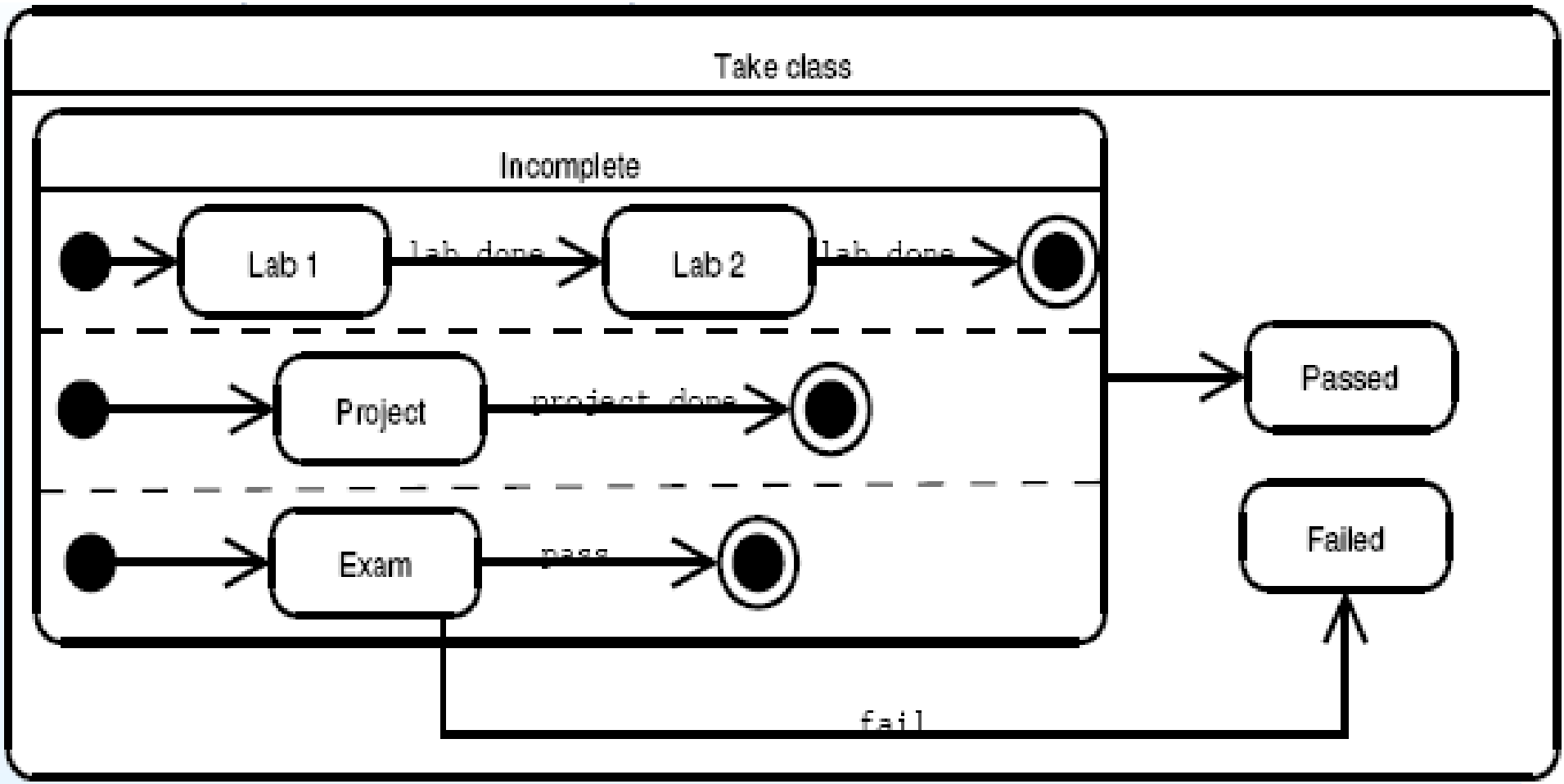
Composited State (1)

- Composite state with under state sequentially active:



Composited State (2)

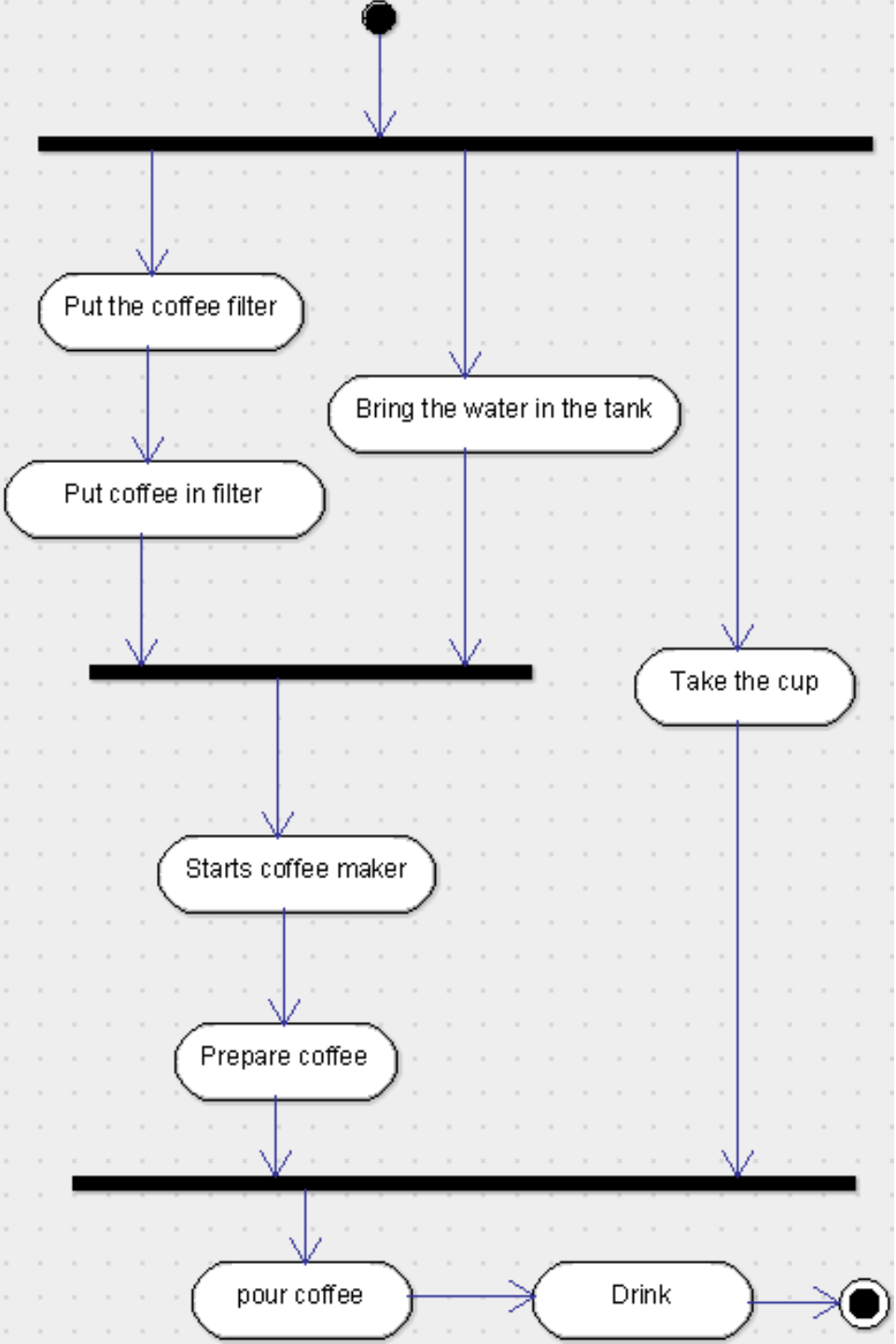
- Composite state with under state parallel active:

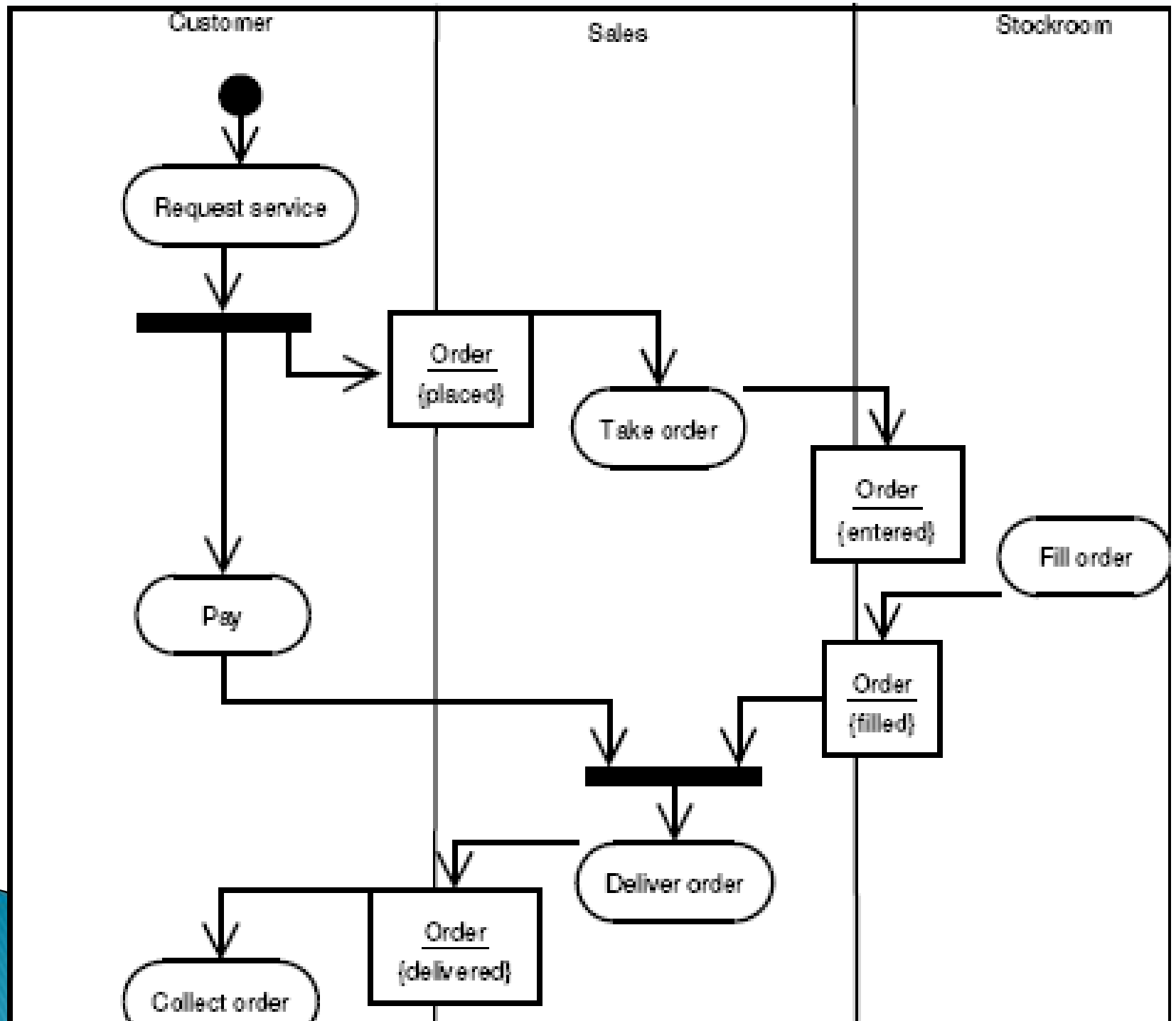


Activity Diagram

- Used to model the dynamics of a **process** or **operation**
- Highlights execution control from one activity to another
- It is attached to:
  - A class (models an use case)
  - A package
  - Implementation of operations

Activity Diagram Example

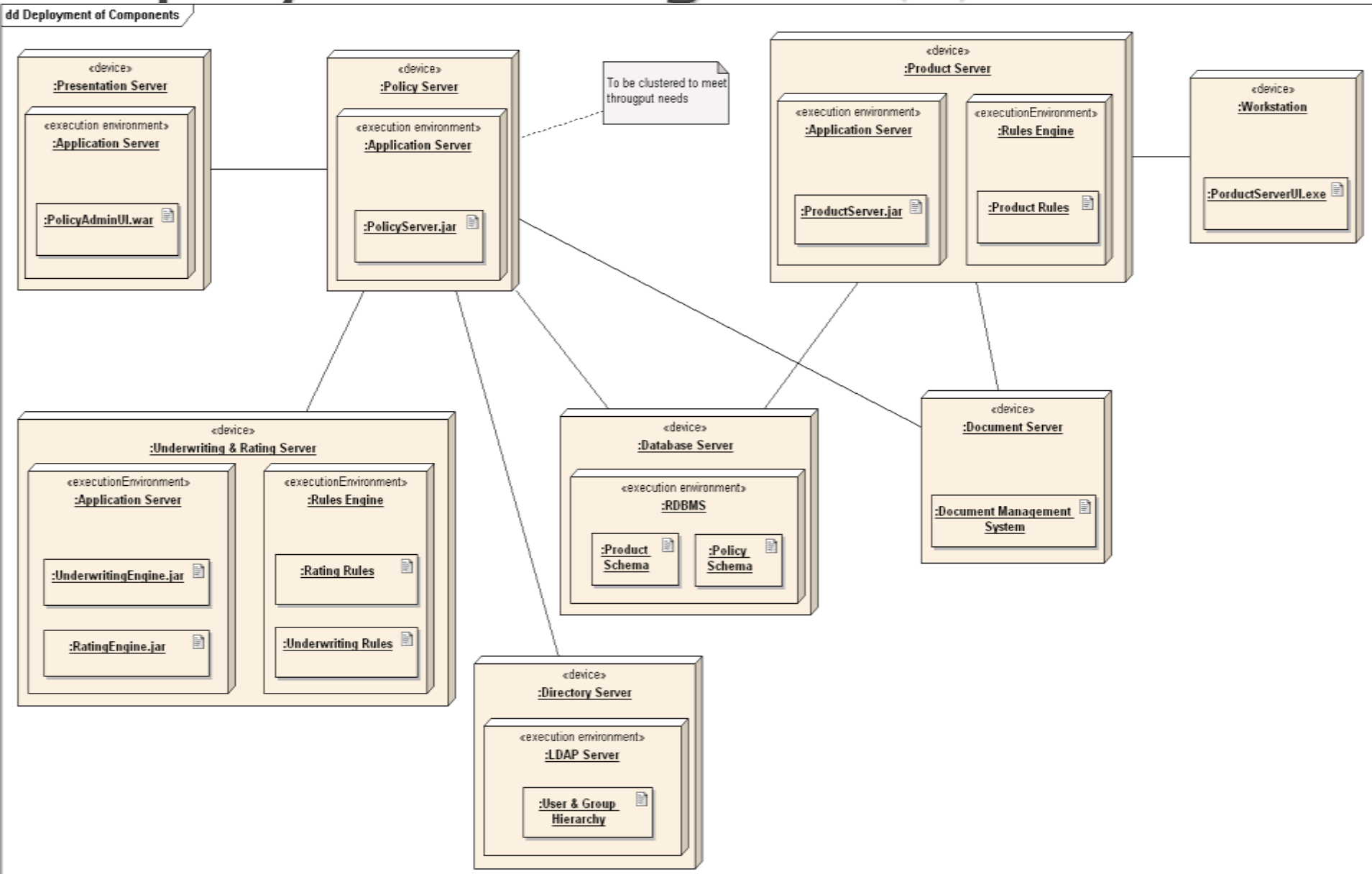



Customer
Sales
Stockroom
Request service
Order
{placed}
Take order
Order
{entered}
Fill order
Pay
Order
{filled}
Deliver order
Order
{delivered}
Collect order

Deployment Diagram

- Model the hardware environment where the project will work
- Example: to describe a web site a deployment diagram will contain hardware components for
  - web server,
  - server applications,
  - server database
- Software components on each of these
  - The web application
  - Database
  - The way in which they are connected: JDBC, REST, RMI

Deployment Diagram (1)



Package Diagram

- Package:
  - It is a logical container for elements between which we have connections
  - Defines a namespace
  - All UML elements can be grouped into packages (most packages are used to group classes)
  - A package may contain subpackages => creates a tree structure (similar to organizing files / folders)

Package Diagrams (2)

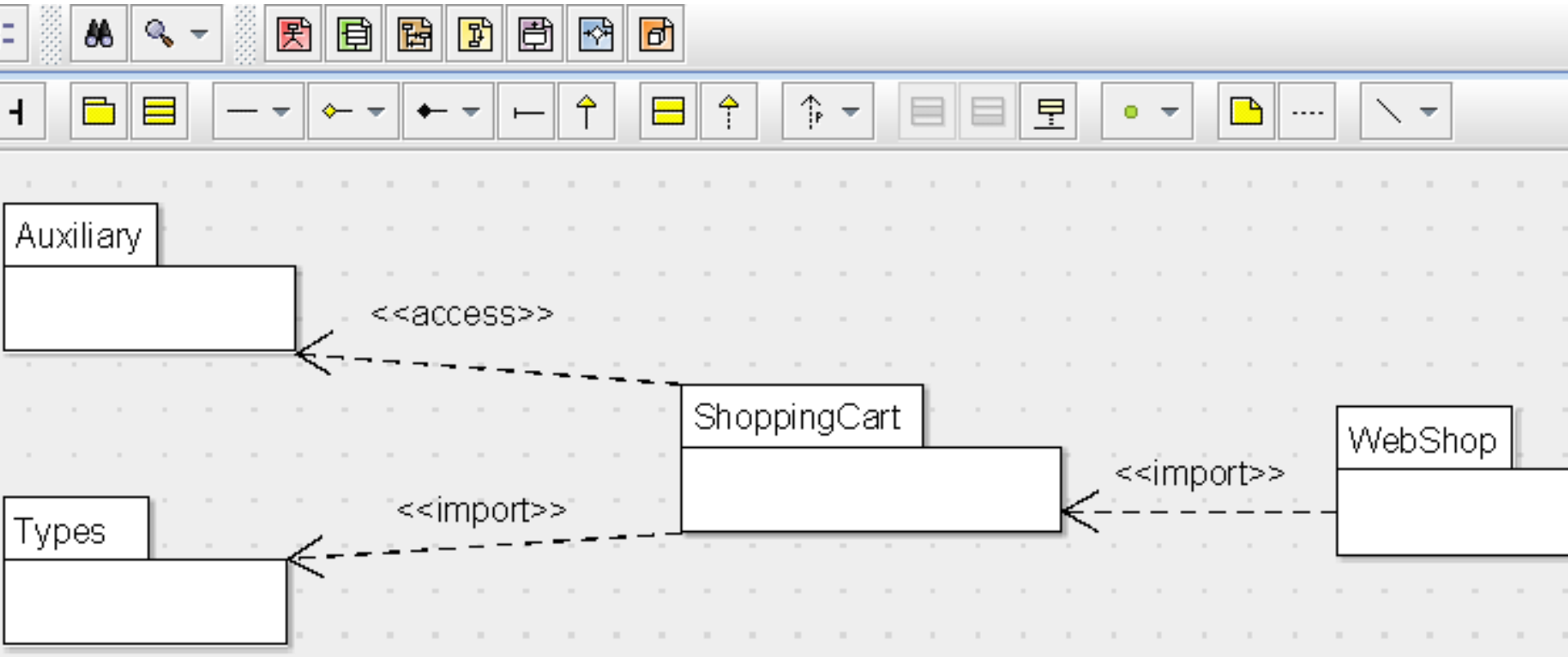
- Relations:
  - dependence <<access>> = privat import
  - dependence <<import>> = public import

- Both relationships allow use of elements found in the destination package by the evidence in the source package without the need of qualifying the names of items in the package destination (similar to import directive in java)

- These types of diagrams are provided in class diagrams

Example

- Elements from Types are imported in ShoppingCart and then are imported by WebShop
- Elements from Auxiliary can be accessed only from ShoppingCart and cannot be referred from WebShop



The utility of Package Diagrams

- Divide large systems into smaller and more manageable subsystems
- Allowing iterative parallel development
- Defining clear interfaces between packages for code reuse (eg. Packages for graphic interface, for enabling connection to BD, etc ...)

Recommendations in making UML diagrams

- Diagrams are **neither too complicated nor too simple**: the goal is **effective communication**
- Give **suggestive names** for components
- Elements arranged so that the **lines do not intersect**
- Try not to show too many kinds of relationships once (avoid complicated diagrams)
- If necessary, **make more diagrams** of the same type

Conclusions

- UML Diagrams:
  - Interactions
  - Behavior
  - Structure

Bibliography

- Ovidiu Gheorghieş, Course 5 IP
- www.uml.org